EITF35: Introduction to Structured VLSI Design

Part 2.1.1: Combinational circuit

Liang Liu
liang.liu@eit.lth.se

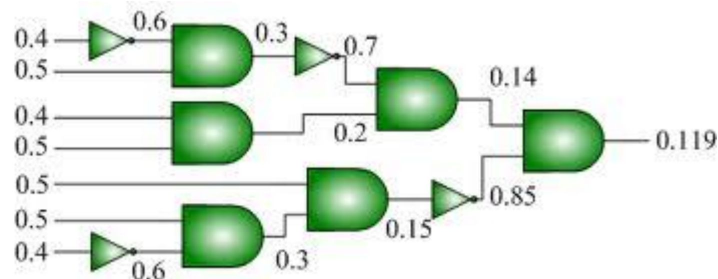
Why Called “Combinational” Circuits?

Combination

- In mathematics a combination is a way of selecting several things out of a larger group
- Select two fruits out of APPLE, PEAR, and ORANGE
- In a combination the **order** of elements is **irrelevant**

Combinational Circuits

- ***time-independent logic***, where the output is a pure function of the present input only.
- the **order** of inputs doesn't matter for the outputs.



‘Digital’- quantization

digital representation

amplitude

time

signal

Data Representation

Unsigned

- Unsigned integer: $$\sum_{i=0}^{n-1} \mathrm{bit}_i 2^i$$

Signed (Two's complement)

- The result of **subtracting the number from 2^{N-1}**
- Inverting all bits and **adding 1**

$$\mathrm{bit}_{n-1}(-2^{n-1}) + \sum_{i=0}^{n-2} \mathrm{bit}_i 2^i$$

$11110100_2 = -12_{10}$

Sign bit (→ leading 1) 2's complement (→ 1110100)

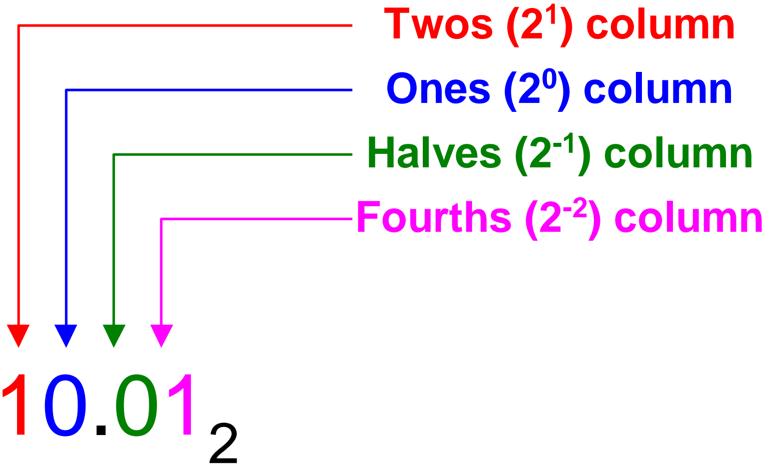
8-bit Signed/Unsigned Integers

	Signed	Binary	Unsigned	
Signed overflow ↑	-128	1000 0000		
	-127	1000 0001		
	...	...		
		1111 1100		
MSB defines sign		1111 1101		
	-2	1111 1110		
	-1	1111 1111		
Signed integers	0	0000 0000	0	
	1	0000 0001	1	
	2	0000 0010	2	
	3	0000 0011	3	
	...	...	...	
	126	0111 1110	126	Unsigned integers
Signed overflow ↓	127	0111 1111	127	
		1000 0000	128	
		1000 0001	129	
		...	...	
		1111 1110	254	
		1111 1111	255	Unsigned overflow ↓

General Fixed-Point Representation

Qm.n notation

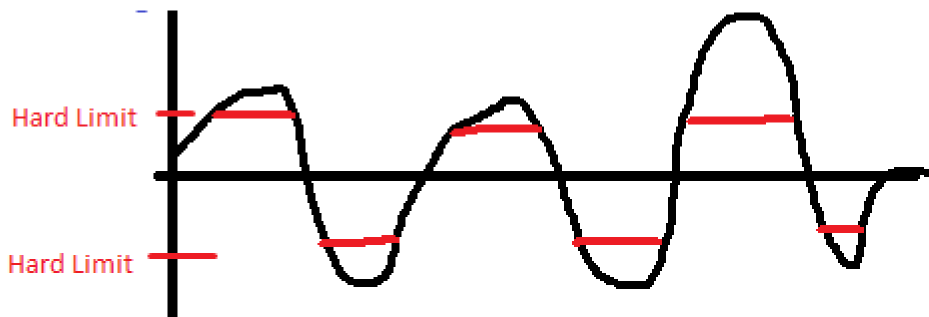
- m bits for integer portion, n bits for fractional portion
- Total number of bits N = m + n + 1, for signed numbers
- Example: 16-bit number (N=16) and Q2.13 format

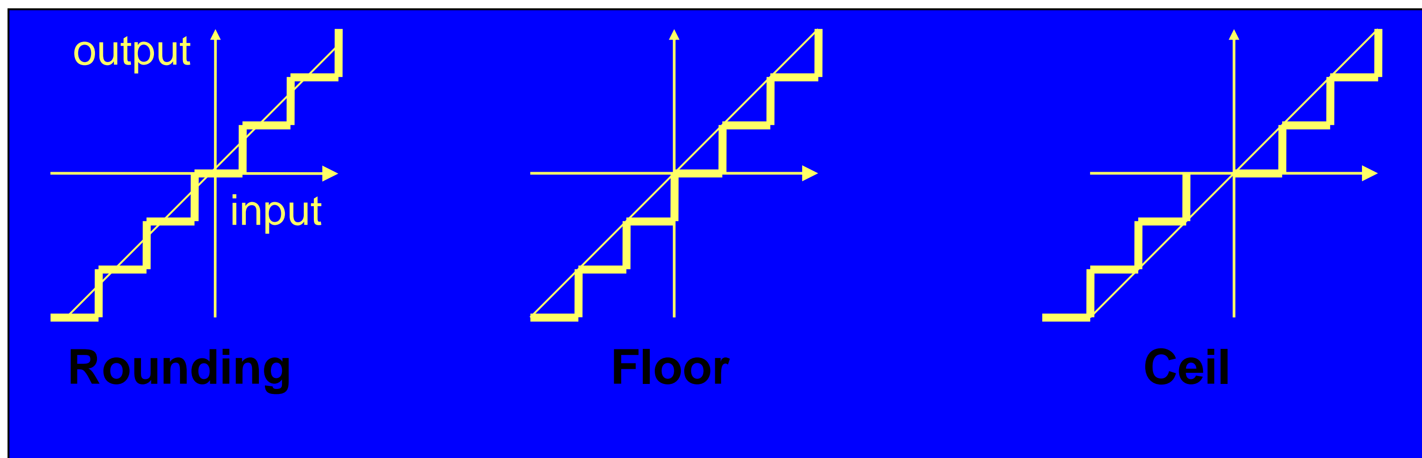


$$= 1\times2^1 + 0\times2^0 + 0\times2^{-1} + 1\times2^{-2}$$

Finite Word-Length Effect

- **Overflow**
  - Saturation
- **Quantization error**
  - Round
  - Truncation





round(0.51)=1   *floor(0.51)=0*   *ceil(0.49)=1*

Will learn more in DSP-Design course

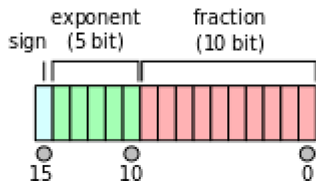
Fixed-Point Design

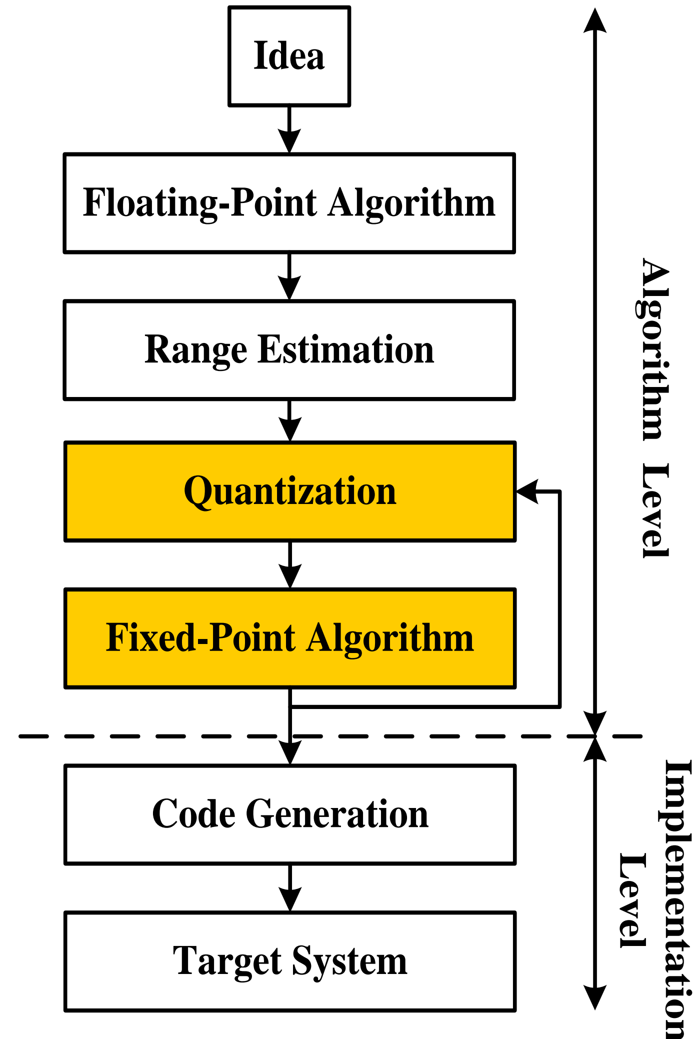
DSP algorithms

- Often developed in floating point
- Later mapped into **fixed point** for digital hardware realization

Fixed-point digital VLSI

- Lower area
- Lower power
- Quantization error & small dynamic range





Optimum Word-Length

- Range Analysis
- Fixed-point Simulation

10^{0}
10^{-1}
10^{-2}
SER
10
15
20
25
SNR (dB)
Floating-point
Fixed-point, 11 bits
Fixed-point, 10 bits
Fixed-point, 9 bits
Fixed-point, 8 bits

Hardware Consumption Analysis

- Complexity analysis
- Quick prototype

Power

8-Bit 46%

Sign-Bit 40%

Area

Sign-Bit 45%

8-Bit 29%

	Sign-Bit		8-Bit	
	Area $[mm^2]$	Power $[mW]$	Area $[mm^2]$	Power $[mW]$
■ Memory	0.018	0.288	0.094	1.543
□ Logic	0.015	0.191	0.038	1.314
Total	0.035	0.479	0.133	2.857

Hardware Consumption Analysis

- Complexity analysis
- Quick prototype

Our greedy search steps
500th generation by GA

Maximum output error: 0, 0.005, 0.01, 0.015, 0.02, 0.025

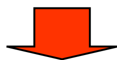
Area: 250, 300, 350, 400, 450

Power: 550, 600, 650, 700, 750, 800

Design Trade-off

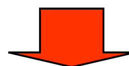
Implement the best HW realization. Best??

Flexibilty
Complexity

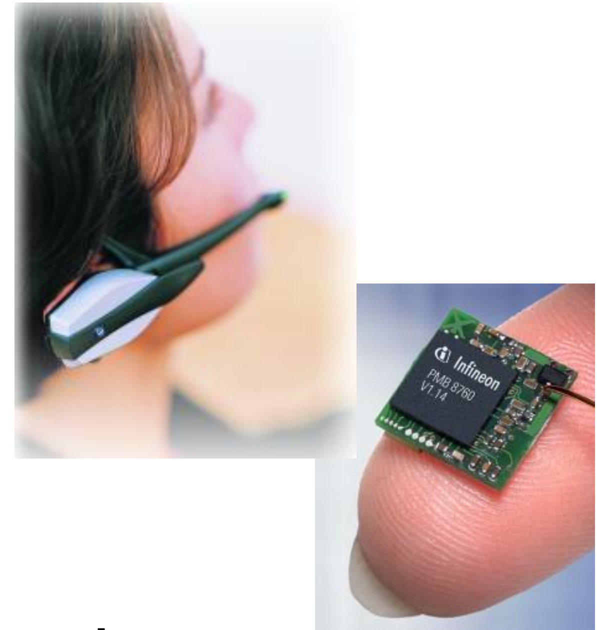


- Processors
- FPGAs

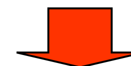
Low power
Low cost
Flexibilty



- Processors
- Dedicated HW



Lower power
Lower cost



- Dedicated HW
- Processors

Design Trade-off

Implement the best HW realization. Best??

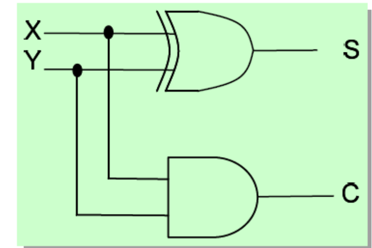
Different applications, different demands...
Thus, ”*just good enough*” is the best in engineering.

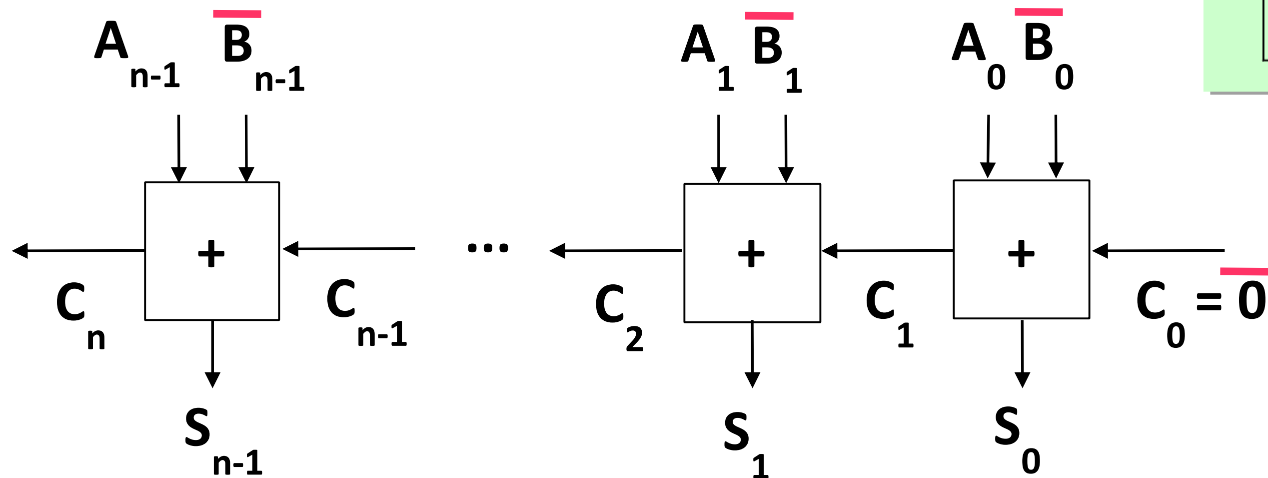
Try to find a BALANCE between effort and cost!

Overview

- Fixed-Point Representation
- **Add/Subtract**
- Multiplication
- Timing&Techniques to Reduce Delay

Add/Subtract (Binary)





- **The HW for sum/difference (S) does NOT care about signed/unsigned**

- **Overflow**
  - Unsigned overflow = C_n
  - Signed overflow = $C_n \oplus C_{n-1}$
  - True sign = $S_{n-1} \oplus$ signed overflow
    $= (A_{n-1} \oplus B_{n-1} \oplus C_{n-1}) \oplus (C_n \oplus c_{n-1}) = A_{n-1} \oplus B_{n-1} \oplus C_n$

Signed Overflow Example

- **4-Bit signed addition**

6+7 = 13, outside [-8..7]

```
        0110
       +0111
C4=0    1101
        C3 = 1
```

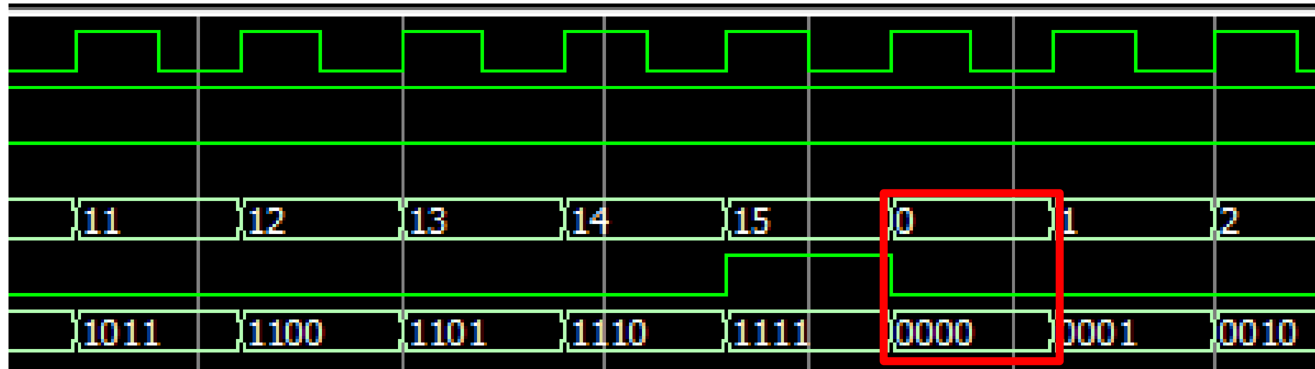
$C_n \oplus C_{n-1} = C_4 \oplus C_3 = 0 \oplus 1 = 1 \Leftrightarrow$
Carry-outs different $\Leftrightarrow$ Signed overflow

$S_{n-1} \oplus$ signed overflow =
$A_{n-1} \oplus B_{n-1} \oplus C_n = A_3 \oplus B_3 \oplus C_4 = 0 \oplus 0 \oplus 0 = 0 \Leftrightarrow$ True sign = Positive/zero

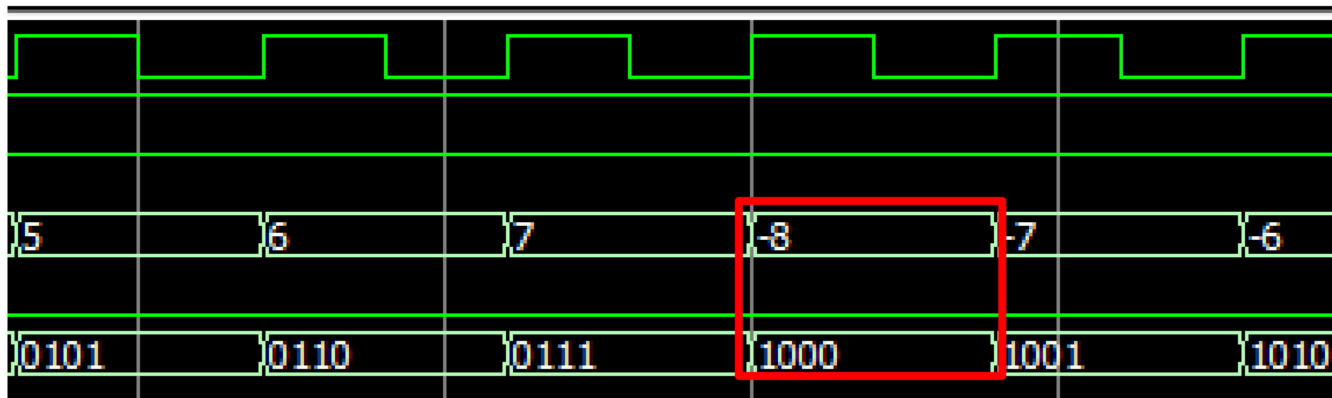
Overflow Check in Hardware?

Overflow in Hardware

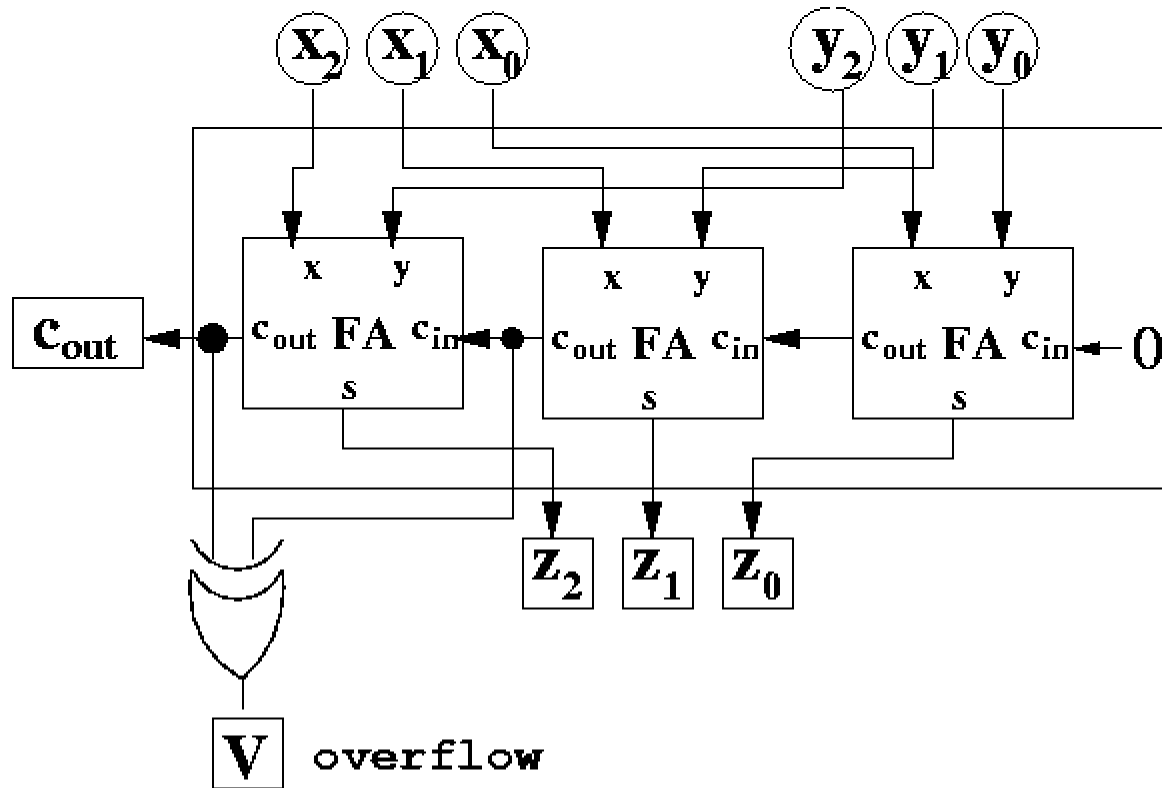
- **Hardware does not take care of the overflow for you**
  - Unsigned



  - Signed



Overflow in Hardware



Saturation or ***wrap-around*** or ***1 more bit***

Two's Complement Signed Extension

- **To add two numbers, we should represent them with the same number of bits: 0100+11100**
  - If we just pad with zeroes on the left:

4-bit		8-bit	
0100	(4)	00000100	(still 4)
1100	(-4)	00001100	(12, not -4)

  - Instead, replicate the MS bit -- the sign bit:

4-bit		8-bit	
0100	(4)	00000100	(still 4)
1100	(-4)	11111100	(still -4)

Decimal Mark in Hardware

- **Matlab aligns the decimal mark automatically**

  1.32+100.2343= 101.5543

- **Hardware does NOT**
  - Decimal mark is just a concept

    01.100+001.01=?

    10001

  - You need to align the decimal mark manually

    001.100+001.010=010.110

Overview

- Fixed-Point Representation
- Add/Subtract
- **Multiplication**
- Timing & Techniques to Reduce Delay

1,6
1,4
1,2
1
0,8
0,6
0,4
0,2
0
Area (mm)
Delay (ns)
Mult
Add

Array Multiplier (unsigned)

```
      1011 * 1110
      0000  (*0 = zero)
    +1011.  (*1 = copy)
   +1011..  (*1 = copy)
  +1011...  (*1 = copy)
  10011010
```

Direct Mapping

- ***Horizontal :*** partial product using AND
- ***Vertical :*** shift-add of partial product

Multiplicand

X_3 X_2 X_1 X_0 Y_0

X_3 X_2 X_1 X_0 Y_1 Z_0

HA FA FA HA

X_3 X_2 X_1 X_0 Y_2 Z_1

Multiplier

FA FA FA HA

X_3 X_2 X_1 X_0 Y_3 Z_2

FA FA FA HA

Z_7 Z_6 Z_5 Z_4 Z_3

Don't Forget ... Signed Multiplication

1 0 1 1	-5x
0 0 1 1	+3

?

1 1 1 1 0 0 0 1	-15

Signed Multiplication

- **Either transform to multiply of non-negative integers:**
  - Record signs and negate any negative factors.
  - Perform unsigned multiplication.
  - Negate product if signs above differ.

```
      0 1 0 1     +5x
      0 0 1 1     +3
      -------
      0 1 0 1
    0 1 0 1
  0 0 0 0
0 0 0 0
0 0 0 1 1 1 1     +15
```

abs(-5)=5
abs(3)=3

-1*1*15=-15

Signed Multiplication

- Or directly perform signed multiplication:
  - Multiplier: positive
  - Multiplicand: positive or negative
  - Sign extend the partial products when adding up

```
        1 0 1 1    -5x
        0 0 1 1    +3
---------------
1 1 1 1 1 0 1 1
1 1 1 1 0 1 1
0 0 0 0 0 0
0 0 0 0 0
---------------
1 1 1 1 0 0 0 1    -15
```

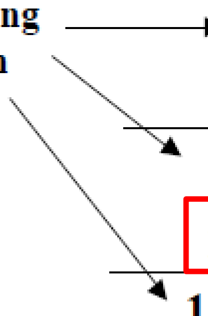
Signed Multiplication

- **Or directly perform signed multiplication:**
  - Multiplier & Multiplicand: positive or negative
  - Sign extend the partial products when adding up
  - Subtract instead of adding last partial product

Ex:

Decimal	Binary	
-5	1011	multiplicand
x -3	x 1101	multiplier
15	00000	partial product
	11011	shifted multiplicand
	111011	partial product
	00000	shifted multiplicand
	1111011	partial product
	11011	shifted multiplicand
	11100111	partial product
	00101	shifted and negated multiplicand
	00001111	product

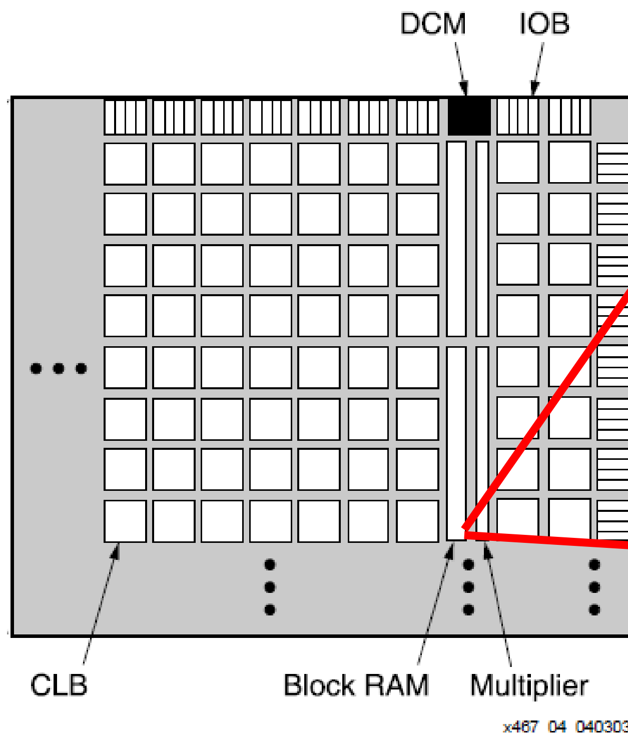
Added bit using sign extension

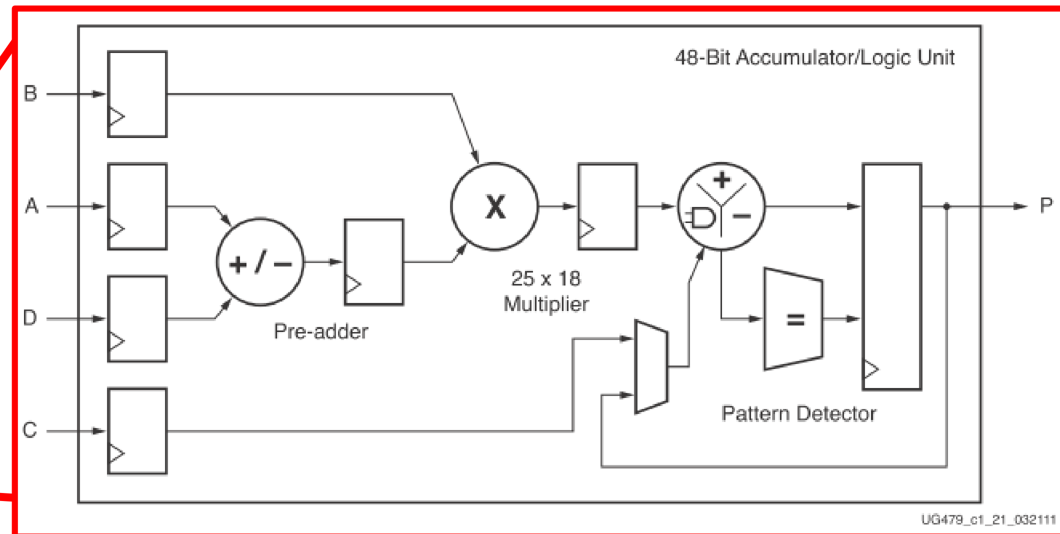


Multiplier in Xilinx FPGA

Embedded DSP48E1

- 25×18 embedded multipliers (**two's-complement multiplier**)
- Using Embedded Multipliers in Artix-7 FPGAs
  http://www.xilinx.com/support/documentation/user_guides/ug479_7Series_DSP48E1.pdf





Multiplier in Xilinx FPGA

Embedded DSP48E1

- 25×18 two's-complement multiplier
- 48-bit accumulator
- Single-instruction-multiple-data (SIMD) arithmetic unit: Dual 24-bit or quad 12-bit add/subtract/accumulate
- Optional pipelining and dedicated buses for cascading

Use suggestions from Xilinx

- Use signed values in HDL source (setting MSB 0 for unsigned)
- Pipeline for performance and lower power, both in the DSP48E1 slice and fabric
- Use the configurable logic block (CLB) carry logic to implement small multipliers, adders, and counters

Multiplier in Xilinx FPGA

Project Settings

General
Simulation
Synthesis
Implementation
Bitstream
IP

Synthesis

Constraints

Default constraint set: constrs_1 (active)

Options

Strategy: Vivado Synthesis Defaults* (Vivado Synthesis 2014)

Description:

Option	Value
-bufg	12
-fanout_limit	10,000
-directive	Default
-fsm_extraction	auto
-keep_equivalent_registers	☐
-resource_sharing*	on
-control_set_opt_threshold	4
-no_lc	☐
-shreg_min_size	3
-max_bram	-1
-max_dsp	-1

More Options

-max_dsp
Maximum number of block DSP allowed in design. (Note -1 means that the tool will choose the max number allowed for the part in question.

OK Cancel Apply

Multiplier in Xilinx FPGA

USE_DSP48 Verilog Example

```
(* use_dsp48 = "yes" *) module test(clk, in1, in2, out1);
```

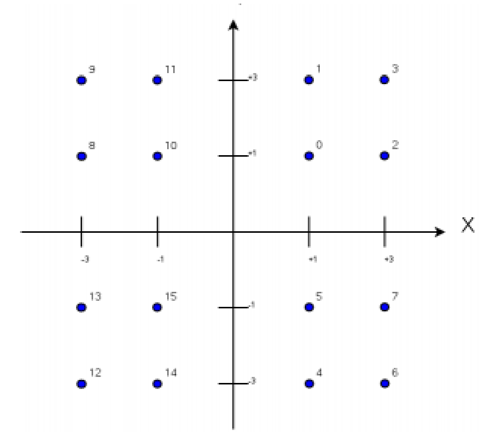
USE_DSP48 VHDL Example

```
attribute use_dsp48 : string;
attribute use_dsp48 of P_reg : signal is "no"
```

```
architecture archi of use_dsp48_example is
  signal s : std_logic_vector (7 downto 0);
  attribute use_dsp48 : string;
  attribute use_dsp48 of s : signal is "yes";
begin
  process (clk)
  begin
    if clk'event and clk = '1' then
       s <= s + a;
    end if;
  end process;
end archi;
```

Constant Multiplication

- **Examples:**
  - **Twiddle factor** in FFTs
  - **Constellation points** in wireless communication



$$y_j = \sum_{k=0}^{n-1} e^{-\frac{2\pi i}{n} jk} x_k$$

- **Software may be not smart enough to optimize**
- **Designer should optimize that multiplications with a small constant is accomplished by shifts & adds**

  Some numerical examples:

  2 (10_2): multiplicand << 1

  3 (11_2): multiplicand << 1 + multiplicand

  5 (101_2): multiplicand << 2 + multiplicand

  255 (11111111_2): ?

  multiplicand << 8 – multiplicand

Different Data representation

6/8 = 0.75

Binary: 0.11
Stochastic : 10111011, p=P(x=1)

[0,1]: p=P(x=1);
[-1,1]: p=2P(x=1)-1

3/10 = 0.3

Binary: ?
Stochastic : 0010010001

Advantagies: Error tolerent

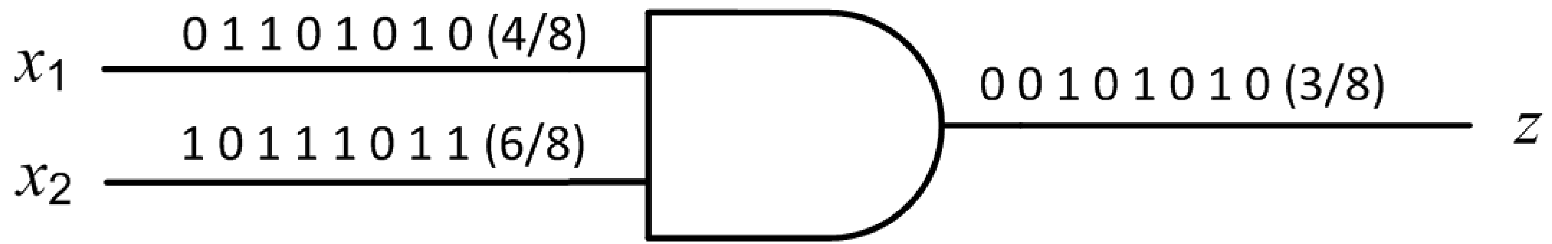
Binary:
3/8: 0.011 → 7/8: 0.111

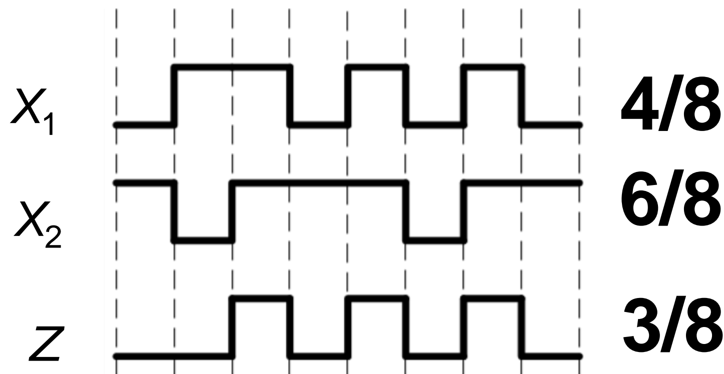
Stochastic:
3/8: 00011010 → 4/8: 10011010

Advantagies: Simple arithemetic

$Z = X_1 \times X_2$

3/8 = 4/8 × 6/8



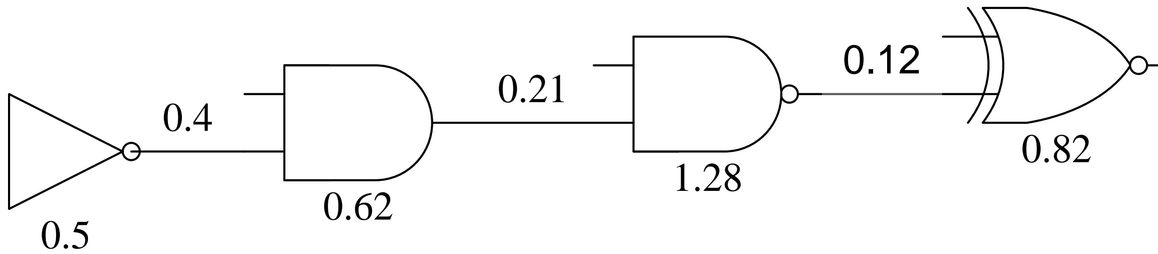


Overview

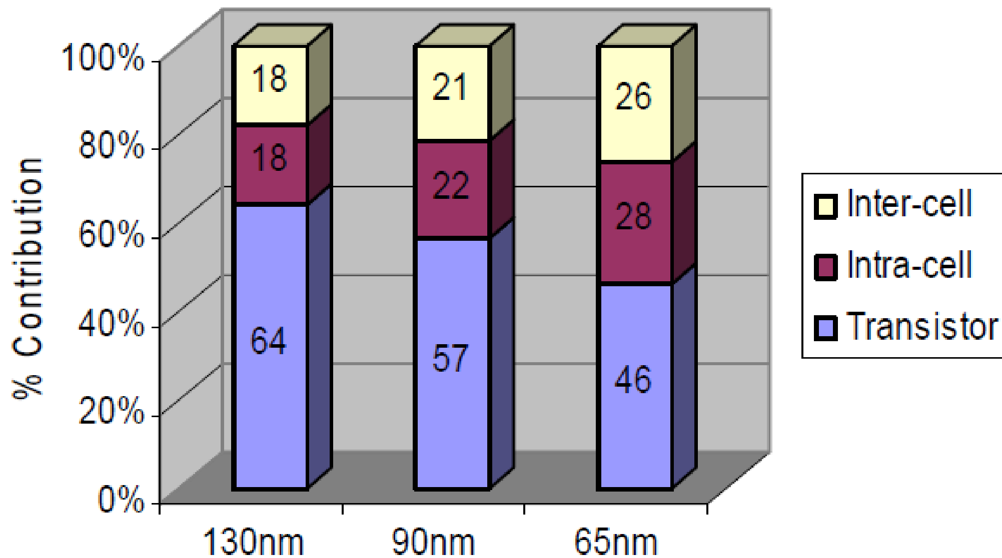
- Fixed-Point Representation
- Add/Subtract
- Multiplication
- Timing & Techniques to Reduce Delay

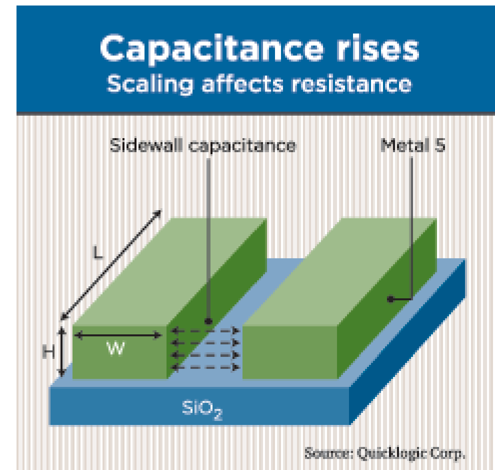
Combinational Circuit Timing

- Path delay = cell delay + net delay



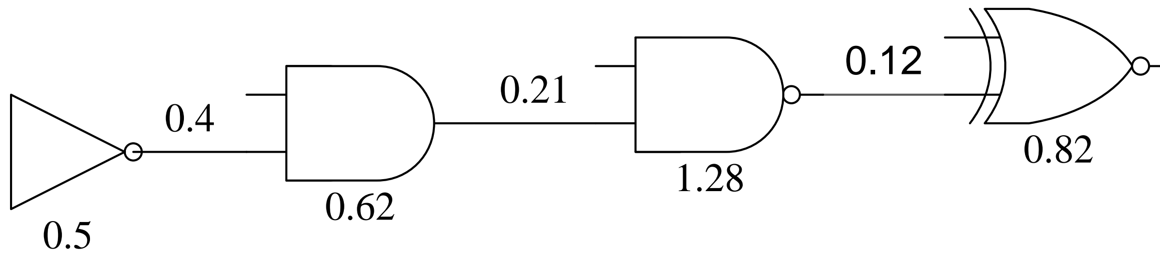
Path Delay = 0.5+0.4+0.62+0.21+1.28+0.12+0.82=3.95 ns



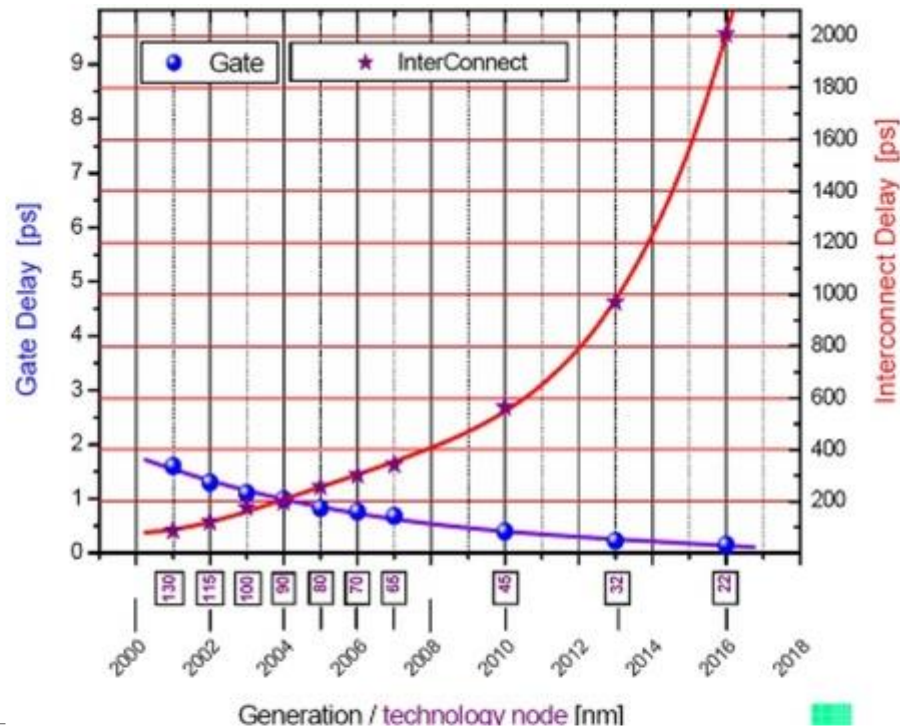


Combinational Circuit Timing

- Path delay = cell delay + net delay

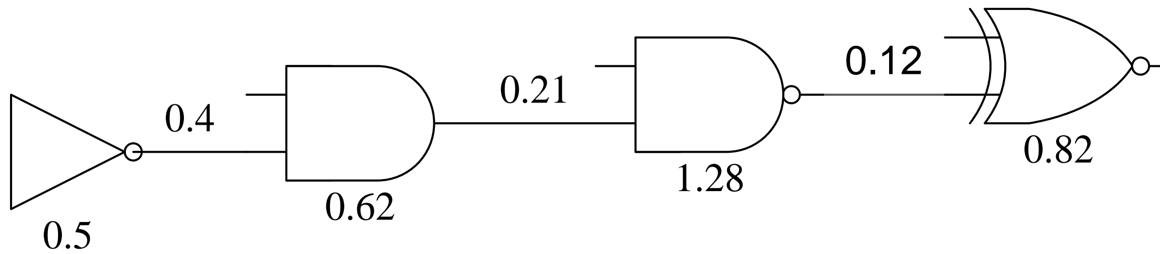


Path Delay = 0.5+0.4+0.62+0.21+1.28+0.12+0.82=3.95 ns

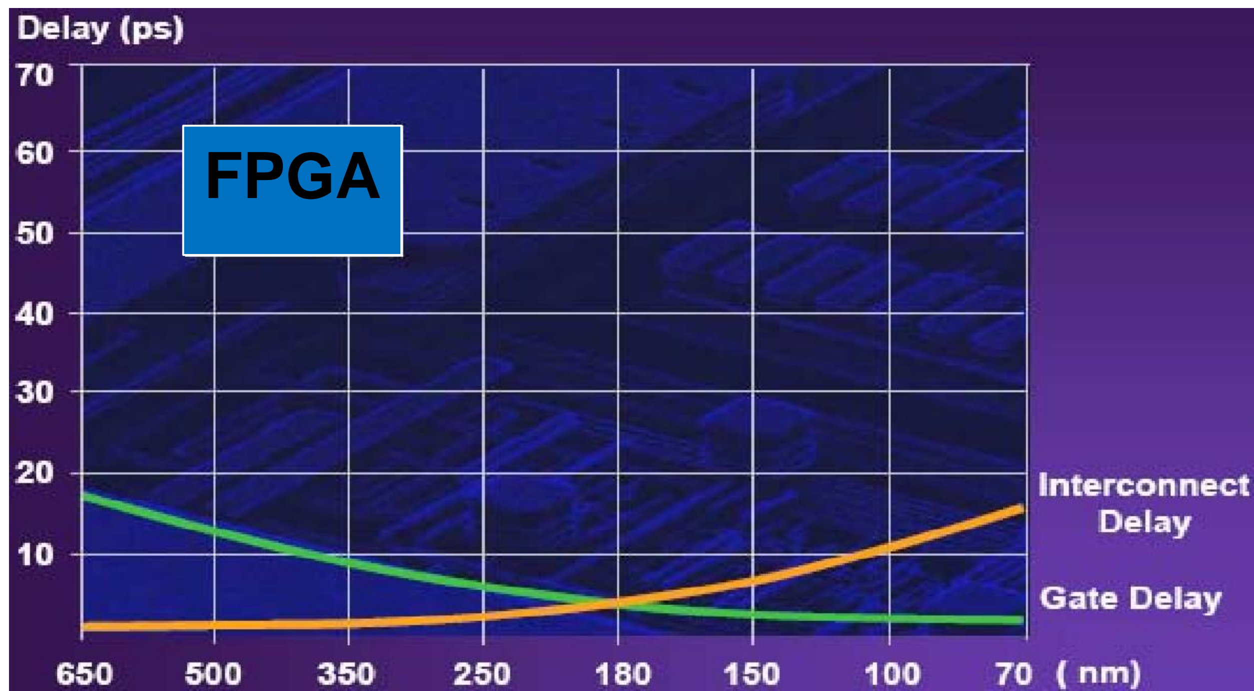


Combinational Circuit Timing

- Path delay = cell delay + net delay

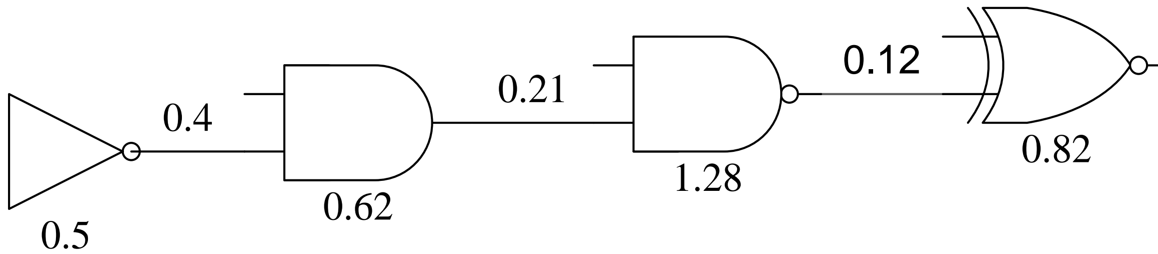


Path Delay = 0.5+0.4+0.62+0.21+1.28+0.12+0.82=3.95 ns



Combinational Circuit Timing

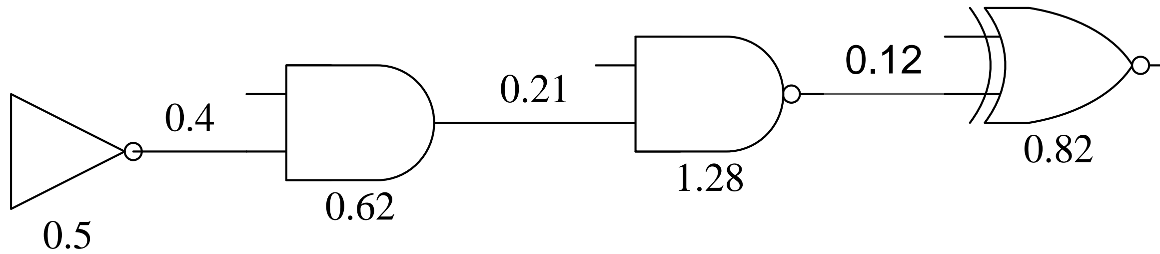
- Path delay = cell delay + net delay



- How to reduce processing delay
  - Reduce cell delay? Standard-cell library (Digital-IC)
  - Reduce net delay? Place & Route (Floor Plan)

Combinational Circuit Timing

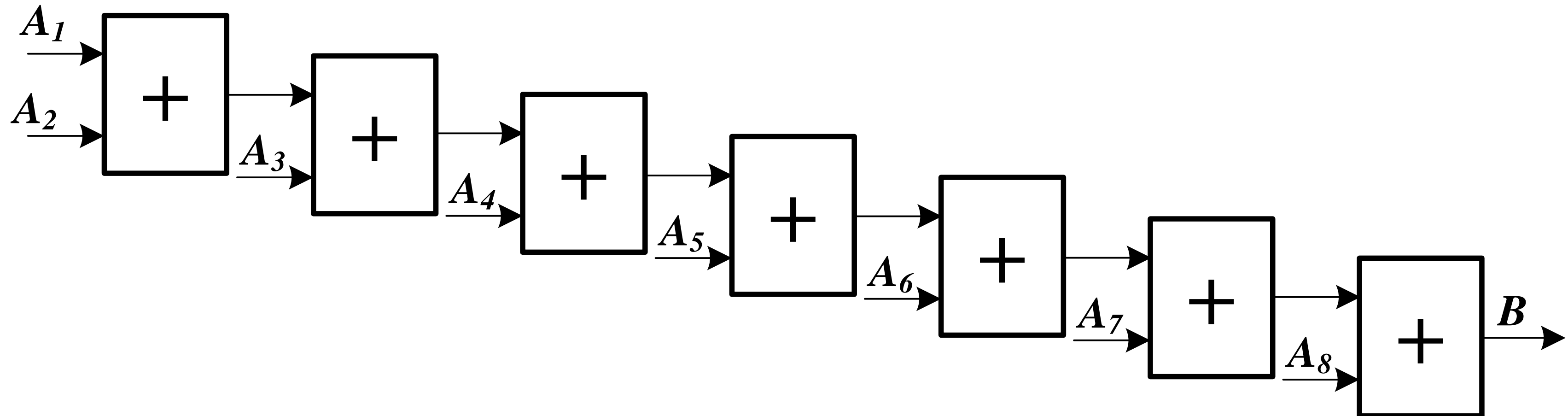
- Path delay = cell delay + net delay



- How to reduce processing delay
  - Reduce cell delay? Standard-cell library (Digital-IC)
  - Reduce net delay? Place & Route
  - **Or we can change the architecture**

Example1: Higher-Level Adder Chain

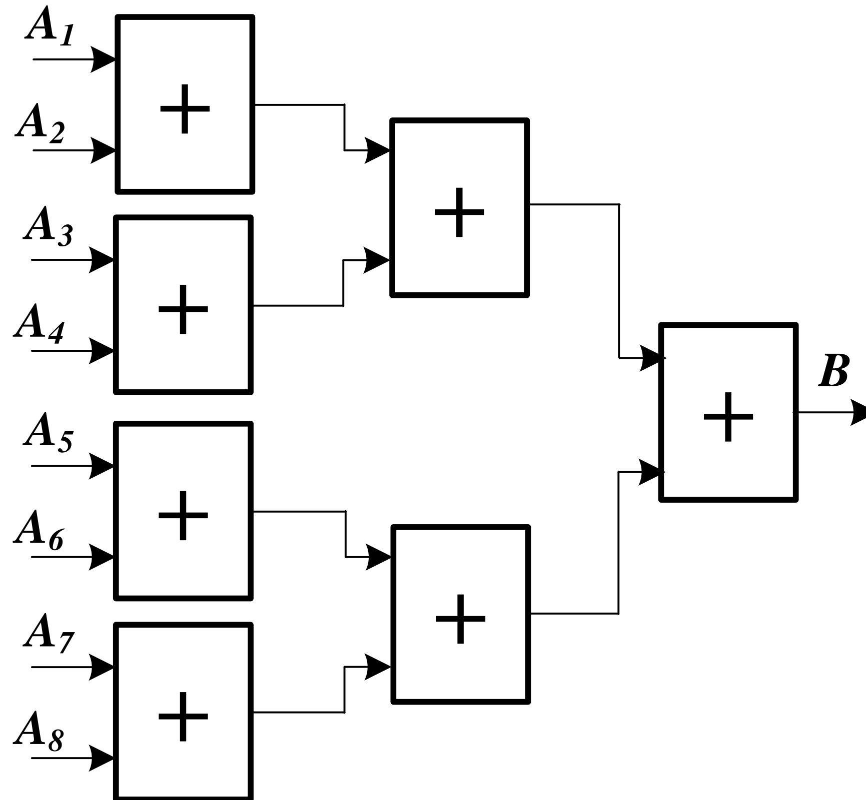
- Calculate: $B = A_1 + A_2 + A_3 + A_4 + A_5 + A_6 + A_7 + A_8$



Cascaded-Chain

Higher-Level

$$B = [(A_1 + A_2) + (A_3 + A_4)] + [(A_5 + A_6) + (A_7 + A_8)]$$



Tree

Cascade vs. Tree

Comparison of n-input adder

- Cascading chain:
  - Area: (n-1) full adder
  - Delay: (n-1)
  - Flexibility: easy to modify (scale)
- Tree:
  - Area: (n-1) full adder
  - Delay: **$\log_2^n$**
  - Flexibility: not so easy to modify

Thanks!